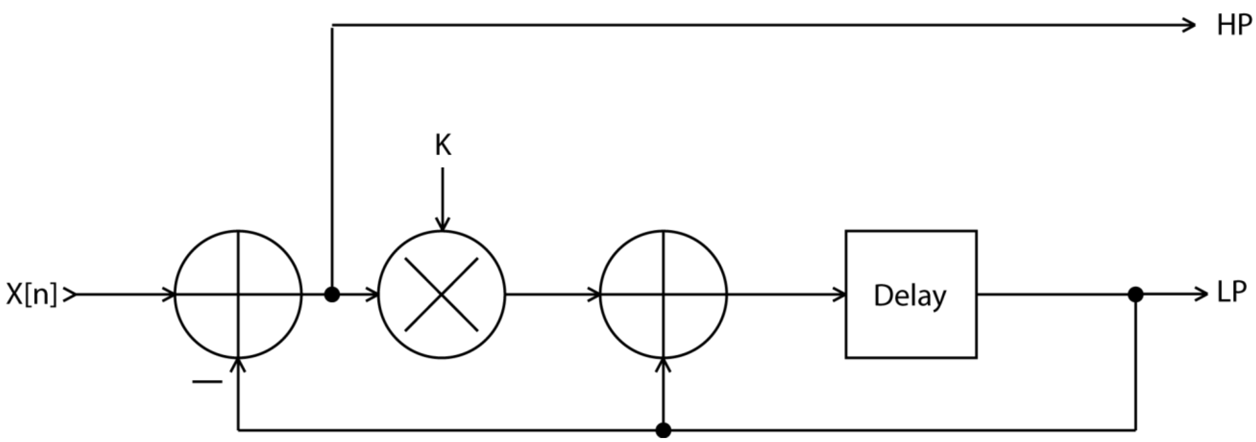
Filter and POT Cheat Sheet

This document will be updated over time to include additional tips.

IIR High Pass and Low Pass Filter



$Y_{HP}[n] = X[n] – Y_{LP}[n-1]$

$Y_{LP}[n] = (X[n] – Y_{LP}[n-1]) * K + Y_{LP}[n-1]$

$K = 1-e^{(-2*pi*Fc/Fs)}$

Fc = -3db point

Fs = sample rate

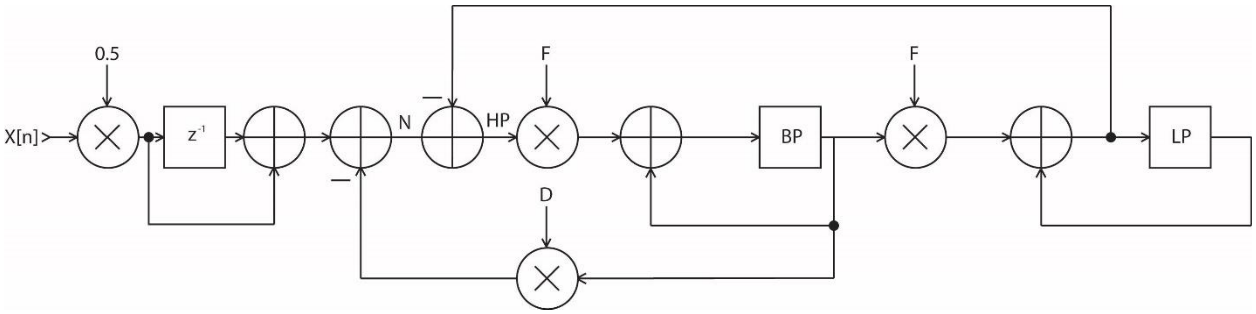
Must calculate HP first as it relies on $Y_{LP}[n-1]$

Code:

```
; User settings
; We want the cutoff frequency to be adjustable between 100Hz and 20KHz @ 48K
.equ fchigh 20000              ; High frequency point
.equ fclow 100                 ; Low frequency point
.equ fs 48000                  ; Sample rate

; Register, may need to change so they do not conflict with other items
.rn temp r0
.rn temp2 r1
.rn lp r2
.rn hp r3

; Constants and equations

.equ e 2.71828183
```

```
.equ pi 3.14159265359

; Calculate K for high and low frequencies
.equ k1high 1-e^(-2*pi*fchigh/fs)
.equ k1low 1-e^(-2*pi*fclow/fs)

; Calculate difference between high and low K
.equ delta k1high-k1low

; Adjust K based on POT0
cpy_cs temp2, pot0_smth        ; Get POT0
multri temp2, delta            ; *delta
addsi acc32, k1low             ; plus base
cpy_cc temp2, acc32            ; save K

; Now filter
cpy_cs temp, in0               ; Read in the input
subs temp, lp                  ; X[n] - Ylp[n]
cpy_cc hp, acc32               ; Save high pass result, can delete this line
if only need low pass
multrr acc32, temp2            ; *K
adds acc32, lp                 ; + Ylp[n]
cpy_cc lp, acc32               ; Save low pass result

cpy_sc out0, hp                ; High pass out on out0
cpy_sc out1, lp                ; Low Pass out on out1
```

State Variable Filter with Band Pass, Notch, Low Pass and High Pass Outputs



F = 2sin(pi*Fc/Fs)

Fc = -3db point

Fs = sample rate

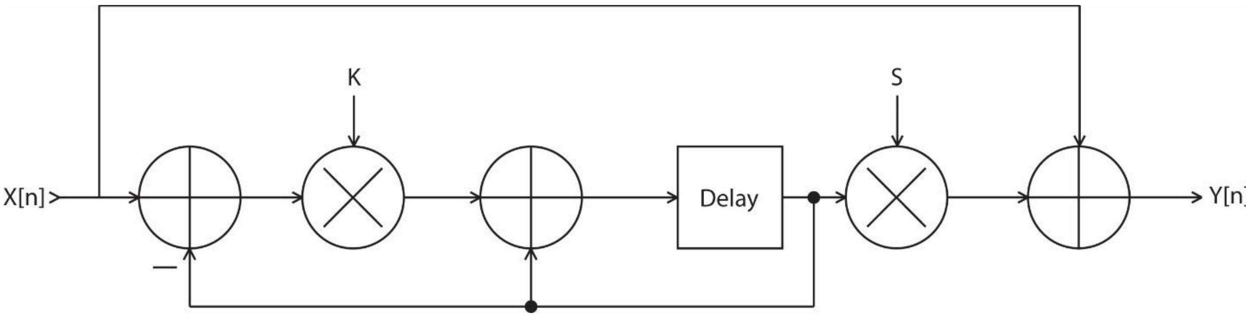
D = 1/Q

Code:

```
; State Variable Filter
; Highpass, lowpass, bandpass and notch all in one structure
; f = 2*sin(pi*Fc/Fs)
; Fc = desired cutoff/center frequency
; Fs = sample rate
;
; d = 1/Q : note that The internal gain of the filter equals Q.
; I.e. if Q = 2, the filter input must be attenuated by 6dB to
; avoid internal clipping
;
; This is an adjustable version and as f ranges 0 to 1 for DC to Fs/2
; we use the pot value directly.
;
; We also use the pot value for d directly but you may want to limit
; this in a real world application


.rn        temp       r0
.rn        temp2      r1
.rn        inlp       r2
.rn        lp         r3
.rn        bp         r4
.rn        notch      r5
.rn        hp         r6
.rn        f          r7
.rn        d          r8

; read in the pot values for f and d
cpy_cs     f, pot0
```

```
        cpy_cs      d, pot1

    ; now the SVF
    ; first a LP FIR with a null at Fs/2 to help make the filter stable
    ; and allow a wider range of coefficients
    cpy_cs      temp, in0                 ; read in0 into temp
    sra         temp, 1                   ; in/2
    cpy_cc      temp, acc32               ; save to temp
    adds        acc32, inlp               ; in/2 + input LP
    cpy_cc      temp2, acc32              ; input to SVF in temp2
    cpy_cc      inlp, temp                ; save in/2 to input LP
    ; now the svf
    multrr      d, bp                     ; Kd * BP
    subs        temp2, acc32              ; input - Kd*BP, this is the notch
    cpy_cc      notch, acc32              ; save notch result
    multrr      f, bp                     ; Kf * BP
    adds        lp, acc32                 ; + LP
    cpy_cc      lp, acc32                 ; save to LP
    subs        notch, acc32              ; Notch - LP is HP
    cpy_cc      hp, acc32                 ; save it
    multrr      f, acc32                  ; Kf * HP
    adds        bp, acc32                 ; + BP
    cpy_cc      bp, acc32                 ; Save to BP

    ; write results to outputs
    cpy_sc      out0, lp
    cpy_sc      out1, bp
    cpy_sc      out2, hp
    cpy_sc      out3, notch
```

Adjustable Low Shelf



K = 1-e^(-2*pi*Fc/Fs)

Fc = Shelf corner

Fs = sample rate

S ranges -1 (cut shelf) to +1 (boost shelf)

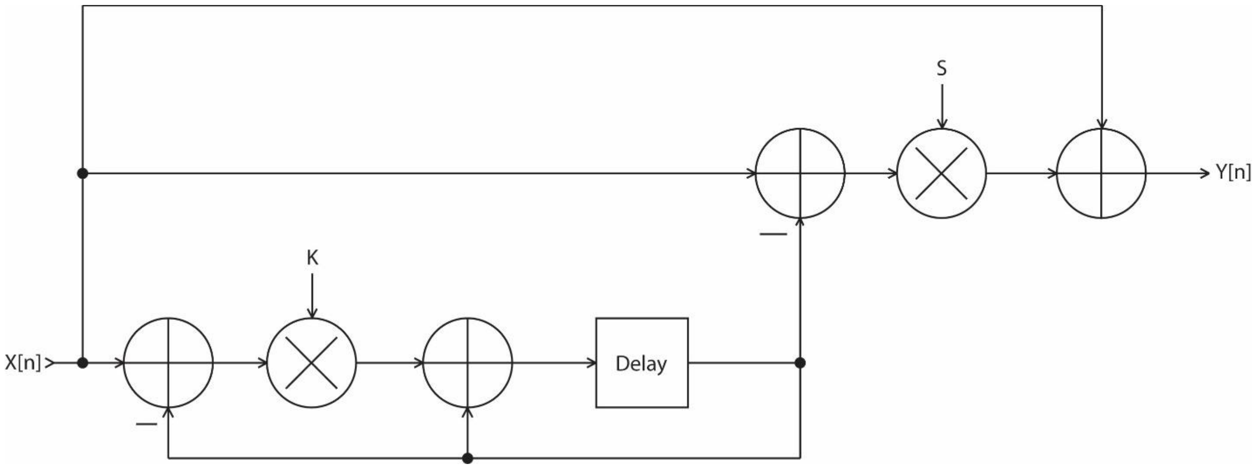
POT0 adjust Fc, POT1 adjusts shelf. Can raise/lower shelf, POT1 at 50% is flat

Code:

```
; Low shelf
; Pot0 adjusts Fc
; Pot1 adjusts shelf level
;
; User settings
; We want the cutoff frequency to be adjustable between 100Hz and 20KHz @ 48K
.equ fchigh 20000              ; High frequency point
.equ fclow 100                 ; Low frequency point
.equ fs 48000                  ; Sample rate

; Register, may need to change so they do not conflict with other items
.rn temp r0
.rn temp2 r1
.rn lp r2
.rn ls r3

; Constants and equations
.equ e 2.71828183
.equ pi 3.14159265359

; Calculate K for high and low frequencies
.equ k1high 1-e^(-2*pi*fchigh/fs)
.equ k1low 1-e^(-2*pi*fclow/fs)

; Calculate difference between high and low K
.equ delta k1high-k1low
```

```
; The filters
; adjust pot1 to range -1.0 to +1.0
cpy_cs temp2, pot1_smth       ; read the pot value
addsi temp2, -0.5             ; shift to -0/5 to +0.5 range
sls acc32, 1                  ; now -1.0 to +1.0 range
cpy_cc temp2, acc32           ; save it
; First low shelf
cpy_cs temp, in0              ; Read in the input
multrr temp2, lp              ; adjust the shelf level from the pot1 scaling
above
adds acc32, temp              ; add the input
cpy_cc ls, acc32              ; Save high shelf result

; Adjust K based on POT0
cpy_cs temp2, pot0_smth       ; Get POT0
multri temp2, delta           ; *delta
addsi acc32, k1low            ; plus base
cpy_cc temp2, acc32           ; save K

; Now low pass
subs temp, lp                 ; X[n] - Ylp[n-1]
multrr acc32, temp2           ; *K
adds acc32, lp                ; + Ylp[n-1]
cpy_cc lp, acc32              ; Save low pass result

cpy_sc out0, ls               ; low shelfout on out0
```

Adjustable High Shelf



$K = 1-e^{(-2*pi*Fc/Fs)}$

Fc = Shelf corner

Fs = sample rate

S ranges -1 (cut shelf) to +1 (boost shelf)

POT0 adjust Fc, POT1 adjusts shelf. Can raise/lower shelf, POT1 at 50% is flat

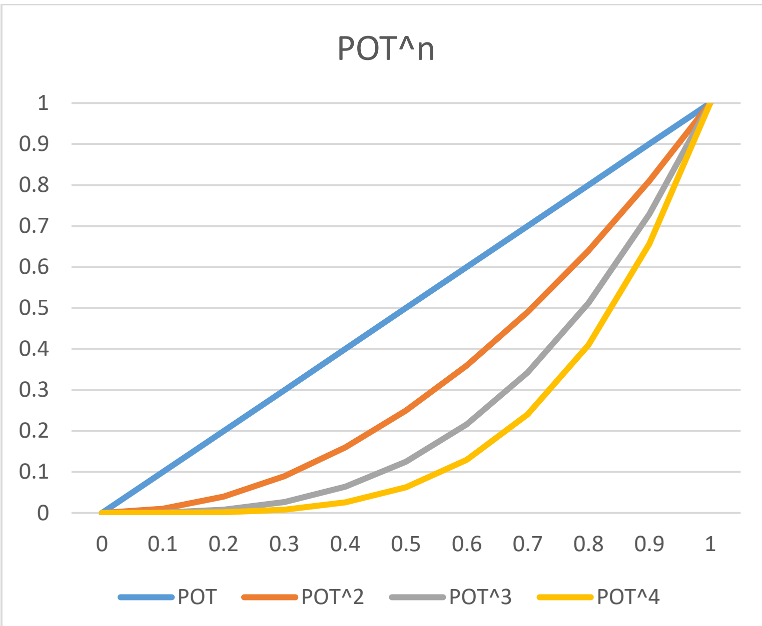
Code:

```
; High shelf
; Pot0 adjusts Fc
; Pot1 adjusts shelf level
;
; User settings
; We want the cutoff frequency to be adjustable between 100Hz and 20KHz @ 48K
.equ fchigh 20000              ; High frequency point
.equ fclow 100                 ; Low frequency point
.equ fs 48000                  ; Sample rate

; Register, may need to change so they do not conflict with other items
.rn temp r0
.rn temp2 r1
.rn lp r2
.rn hs r3

; Constants and equations
.equ e 2.71828183
.equ pi 3.14159265359

; Calculate K for high and low frequencies
.equ k1high 1-e^(-2*pi*fchigh/fs)
```

```
.equ k1low 1-e^(-2*pi*fclow/fs)

; Calculate difference between high and low K
.equ delta k1high-k1low

; The filters
; adjust pot1 to range -1.0 to +1.0
cpy_cs temp2, pot1_smth       ; read the pot value
addsi temp2, -0.5             ; shift to -0/5 to +0.5 range
sls acc32, 1                  ; now -1.0 to +1.0 range
cpy_cc temp2, acc32           ; save it
; First high shelf
cpy_cs temp, in0              ; Read in the input
subs temp, lp                 ; Yhp[n] = X[n] -Ylp[n-1]
multrr temp2, acc32           ; adjust the shelf level from the pot1 scaling
above
adds acc32, temp              ; add the input
cpy_cc hs, acc32              ; Save high shelf result

; Adjust K based on POT0
cpy_cs temp2, pot0_smth       ; Get POT0
multri temp2, delta           ; *delta
addsi acc32, k1low            ; plus base
cpy_cc temp2, acc32           ; save K

; Now low pass
subs temp, lp                 ; X[n] - Ylp[n-1]
multrr acc32, temp2           ; *K
adds acc32, lp                ; + Ylp[n-1]
cpy_cc lp, acc32              ; Save low pass result

cpy_sc out0, hs               ; High shelf out on out0
```

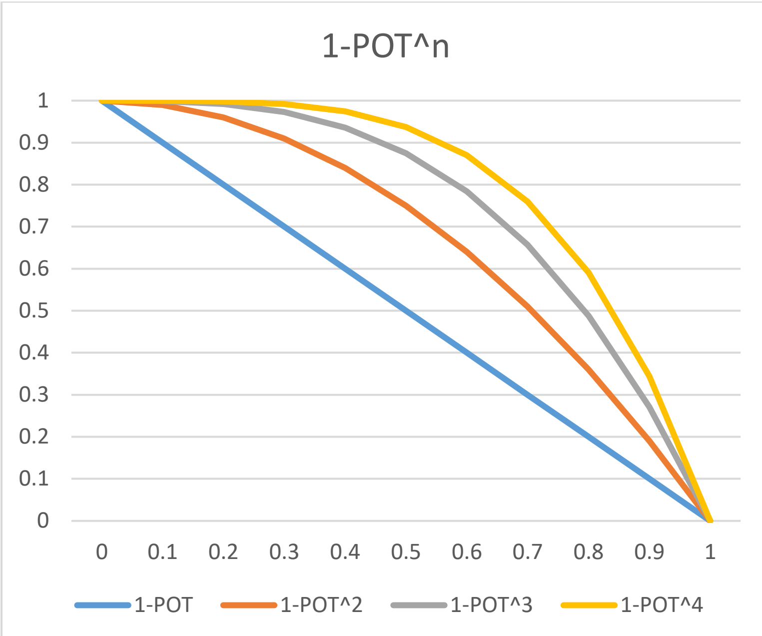
POT Curves

The POT inputs in FXCore are linear but at times a non-linear curve may be desired, this can be accomplished a number of ways depending on the desired curve. Horizontal axis is POT position.



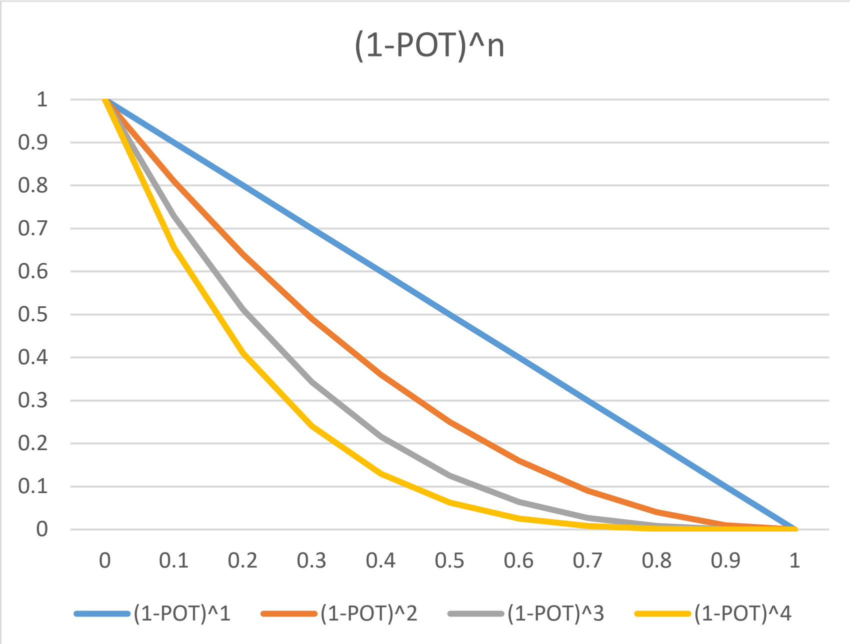
Code:

```
; POT^3 example
;

.rn temp r0

cpy_cs    temp, pot0_smth     ; pot0 in temp
multrr    temp, temp          ; pot0^2 in acc32
multrr    temp, acc32         ; pot^3 in acc32

; note you can continue the "multrr    temp, acc32" instructions
; as many times as required giving: pot0^4, pot0^5, etc.

cpy_cs    temp, in0           ; read input
multrr    temp, acc32         ; using pot0^3 as a volume control
cpy_sc    out0, acc32         ; output result
```



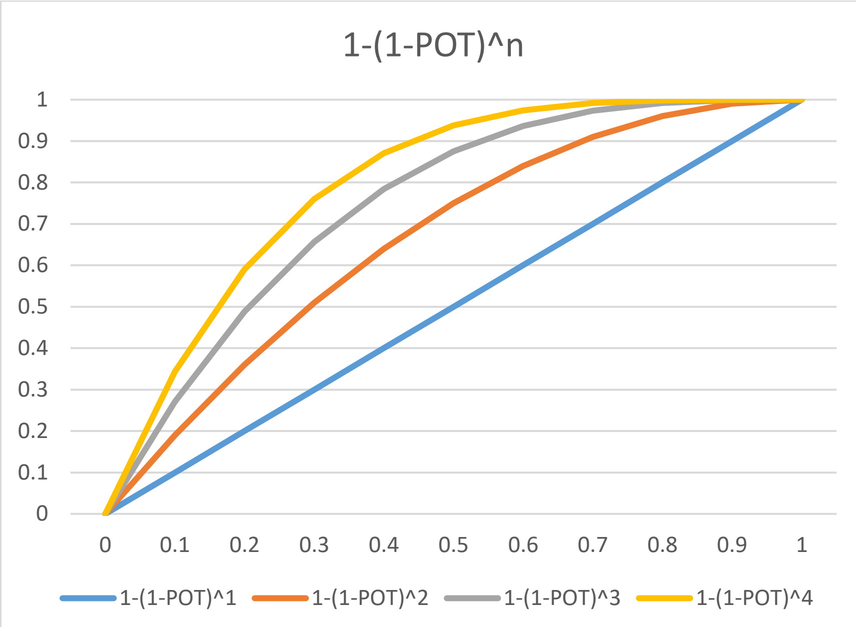
Code:

```
; 1-POT^3 example
;

.rn temp r0
.rn temp2 r1

cpy_cs     temp, pot0_smth       ; pot0 in temp
multrr     temp, temp            ; pot0^2 in acc32
multrr     temp, acc32           ; pot^3 in acc32

; note you can continue the "multrr    temp, acc32" instructions
; as many times as required giving: pot0^4, pot0^5, etc.

cpy_cc     temp, acc32           ; save the result
wrdld      acc32, 0x7fff         ; put almost 1.0 into acc32
ori        acc32, 0xffff         ; don't forget the LSBs
subs       acc32, temp           ; 1-pot0^3 in acc32
cpy_cs     temp, in0             ; read input
multrr     temp, acc32           ; using 1-pot0^3 as a volume control
cpy_sc     out0, acc32           ; output result
```



Code:

```
; (1-POT)^3 example
;

.rn temp r0
.rn temp2 r1

cpy_cs     temp, pot0_smth      ; pot0 in temp
wrdld      acc32, 0x7fff        ; put almost 1.0 into acc32
ori        acc32, 0xffff        ; don't forget the LSBs
subs       acc32, temp          ; 1-pot0 in acc32
cpy_cc     temp, acc32          ; copy to temp
multrr     temp, temp           ; (1-pot0)^2 in acc32
multrr     temp, acc32          ; (1-pot0)^3 in acc32

; note you can continue the "multrr    temp, acc32" instructions
; as many times as required giving: pot0^4, pot0^5, etc.

cpy_cs     temp, in0            ; read input
multrr     temp, acc32          ; using (1-pot0)^3 as a volume control
cpy_sc     out0, acc32          ; output result
```



Code:

```
; 1-(1-POT)^3 example
;

.rn temp r0
.rn temp2 r1

cpy_cs     temp, pot0_smth     ; pot0 in temp
wrdld      acc32, 0x7fff       ; put almost 1.0 into acc32
ori        acc32, 0xffff       ; don't forget the LSBs
subs       acc32, temp         ; 1-pot0 in acc32
cpy_cc     temp, acc32         ; copy to temp
multrr     temp, temp          ; (1-pot0)^2 in acc32
multrr     temp, acc32         ; (1-pot0)^3 in acc32

; note you can continue the "multrr    temp, acc32" instructions
; as many times as required giving: pot0^4, pot0^5, etc.

cpy_cc     temp, acc32         ; save the result
wrdld      acc32, 0x7fff       ; put almost 1.0 into acc32
ori        acc32, 0xffff       ; don't forget the LSBs
subs       acc32, temp         ; 1-(1-pot0)^3 in acc32
cpy_cs     temp, in0           ; read input
multrr     temp, acc32         ; using 1-pot0^3 as a volume control
cpy_sc     out0, acc32         ; output result
```

Experimental Noize Inc. reserves the right to make changes to, or to discontinue availability of, any product or service without notice.

Experimental Noize Inc. assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using any Experimental Noize Inc. product or service. To minimize the risks associated with customer products or applications, customers should provide adequate design and operating safeguards.

Experimental Noize Inc. make no warranty, expressed or implied, of the fitness of any product or service for any particular application.

In no even shall Experimental Noize Inc. be liable for any direct, indirect, consequential, punitive, special or incidental damages including, without limitation, damages for loss and profits, business interruption, or loss of information arising out of the use or inability to use any product or document, even if Experimental Noize Inc. has been advised of the possibility of such damage.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Experimental Noize Inc. products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications"). Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Experimental Noize Inc. products are not designed nor intended for use in military or aerospace applications or environments. Experimental Noize Inc. products are not designed nor intended for use in automotive applications.

Experimental Noize Inc.

Scottsdale, AZ USA

www.xnoize.com

sales@xnoize.com